BETA BASIC NEWSLETTER No. 15

****
SEARCHING FOR SOMETHING WHICH ISN'T

From time to time I find I want to look for the first byte in a memory area which isn't a zero, or the first character in a string which isn't a space, or perhaps any character greater than 127. (One example might be compressing a simple screen, when most data will be zeros.) I always think it is a pity that I cannot use INSTRING to speed things up. At one time I was going to upgrade INSTRING to allow such searches, but it got a bit complicated. However, below I provide a simple machine code routine that can look through a memory area, comparing each byte with a particular value (called "comp" in line 40 - this can be anything between 0 and 255). The POKE in line 50 controls the kind of search, which can be for "not equal to comp", "equal to comp" (equivalent to a 1-character INSTRING), "less than comp", or "greater than or equal to comp". By altering the value of comp by 1, you can obtain the equivalents of "less than or equal to" and "greater than". Lines 10 to 30 just POKE the code into the UDG area and need only be used once. Another location could be used as an alternative to USR "a" - the code is relocatable. Lines 60 and 70 determine the length and start address of the search - here the search is of A$, but any part of memory could be dealt with. The result of calling the machine code (the value of USR) is zero if the condition is never satisfied, or it is the position in the searched area where the condition was true and the search finished, i.e. 1 for the first byte, 2 for the second, etc. I will leave you to tailor the routine to particular applications and make it more user-friendly!

```
10 FOR n=USR "a" TO USR "a"+24
20   READ a
     POKE n,a
   NEXT n
30 DATA 1,0,0,33,0,0,80,89,120,177,200,11,126,35,254,0,0,
   246,235,167,237,66,68,77,201
40 POKE USR "a"+15,comp
50 POKE USR "a"+16,40
   REM 40 for <>, 32 for =, 48 for <, 56 for >=
60 DPOKE USR "a"+1,LEN a$
70 DPOKE USR "a"+4,LENGTH(0,"a$")
80 PRINT USR USR "a"
```

****
POINTER CONTROL

This is (I think!) the Kempston Mouse Pointer program sent in long ago by Garry Rowland, with my modifications to PROC MOUSE to allow keyboard rather than mouse control of the pointer. Remove lines 1010 and 1011, and de-REM line 1015, if you want to use a Kempston Mouse. My additions let keys 5-8 move the pointer; 1 sets "DRAW", 2 sets "unDRAW" and 3 exits the loop.

```
10 LET x1=128,y1=88,b=1
20 CLS
   KEYWORDS 0
   CSIZE 8
```

```
  30 pointerinit
  40 DO
       pointeron
  50   DO
         mouse
       LOOP WHILE x1=mx AND y1=my AND b=0
  60   pointeroff
  70   IF b=2 THEN PLOT OVER 2;mx,my
         PLOT OVER 1;mx,my
  80   IF b=1 THEN PLOT OVER 2;mx,my
  90   LET mx=x1,my=y1
 100 LOOP UNTIL b=3
 110 STOP

1000 DEF PROC mouse
1010   LET k=INSTRING(1,"5678123",INKEY$)
1011   ON k
         LET x1=x1-1
         LET y1=y1-1
         LET y1=y1+1
         LET x1=x1+1
         LET b=1
         LET b=2
         LET b=3
1015  REM LET x1=IN 64479,y1=IN 65503*.68,b=3-AND(3,IN 64223)
1020 END PROC

1030 DEF PROC pointeron
1040   GET g$,mx,my
1050   PLOT OVER 2;mx,my," ALTER "
1060   PLOT OVER 1;mx,my," BLANK "
1070 END PROC

1080 DEF PROC pointeroff
1090   PLOT OVER 0;mx,my,g$
1100 END PROC

1110 DEF PROC pointerinit
1120   RESTORE 1200
1130   LOCAL a,d,L
1140   LET a=USR "a"
1150   FOR L=0 TO 15
1160     READ d
         POKE a+L,d
1170   NEXT L
1180   mouse
       LET mx=x1,my=y1
1190 END PROC

1200 DATA 64,224,240,248,252,248,248,88,64,160,144,136,132,
     136,168,88
```

Lines 1050 and 1060 contain UDG "A" and UDG "B", which appear as keywords in the listing but, since the program sets KEYWORDS 0, appear as UDGs when the program is run. The DATA statement defines their shapes as two forms of an arrow pointer. The ideas in the listing above could form a useful part of a drawing or menu-selection program.

****

FILE ERASER

This contribution is from Albert F. Olivera (Gibraltar). He writes:

"When developing programs it is usual to make frequent saves as writing progresses. The reult is a cartridge full of outdated versions, and it becomes a boring task eraing files one by one. I enclose a short program which should make this chore much easier. You will see I have made use of the "catalogue to a string" procedure you printed in Newsletter No.9."

The program reads the cartridge or Opus Discovery disc catalogue into c$, and prints it in two columns. Then a FLASHing bar (in the original it was BRIGHT, but this didn't show up on my monitor) is moved to each file name in turn by pressing, say, ENTER. When you press DELETE, the current name is removed from the screen and added to a list of file names to be ERASEd, kept in z$. You keep doing this until you are happy (the cursor bar will wrap to the top of the catalogue if required) and then press STOP. All the files in z$ will then be erased, and you can go on to another cartridge. Notes: the quotes in lines 80, 110 and 140 should contain 10 spaces. The " STOP "in line 120 must be typed as symbol-shift/A. The form of the ERASE command in line 180 was originally ERASE "m";1;z$(n TO n+9) but I had to alter it on my Discovery.

```
 10 CLS
    CLEAR #
 20 PRINT AT 6,10;"ERASER";
 30 PRINT #0;"INSERT CARTRIDGE"
    PAUSE 0
    INPUT ;
 40 PRINT AT 10,7;"READING CATALOGUE"
 50 cat_to c$
 60 printcat
 70 LET file=1,last=(LEN c$-16)/11,oldfile=last,z$=""
 80 DO
      PRINT AT INT ((file-1)/2),16*NOT (file-INT (file/2)*2)
      ; FLASH 1; OVER 1;"          "
 90   LET oldfile=file
100   GET a$
110   IF a$=CHR$ 12 THEN
        LET z$=z$+c$(file*11+2 TO file*11+11),c$(file*11+2
        TO file*11+11)="          "
        printcat
120 EXIT IF a$=" STOP "
130   LET file=file+1
      IF file>last THEN
        LET file=1
140   PRINT AT INT ((oldfile-1)/2),16*NOT (oldfile-INT (oldf
      ile/2)*2); FLASH 0; OVER 1;"          "
150 LOOP
160 CLS
    FOR n=1 TO LEN z$ STEP 10
170   PRINT "ERASING ";z$(n TO n+9)
180   ERASE z$(n TO n+9)
190 NEXT n
200 PRINT #0;"DONE! HIT ANY KEY"
    PAUSE 0
    RUN
```

```
1000 DEF PROC cat_to REF a$
1010   LET a$=""
1020   DPOKE DPEEK(23631)+15,3973
1030   DPOKE 23643,LENGTH(0,"a$")
1040   CAT #3;1
1050   DPOKE LENGTH(0,"a$")-2,DPEEK(23643)-LENGTH(0,"a$")
1060   DPOKE DPEEK(23631)+15,64423
1070 END PROC

2000 DEF PROC printcat
2010   CLS
2020   FOR n=13 TO LEN c$-10 STEP 11
2030     PRINT c$(n TO n+9),
2040   NEXT n
2050   PRINT ''c$(LEN c$-1 TO LEN c$)
2060   PRINT #0;"ANY KEY=Cursor; DELETE Unwanted Files; STOP
       Erases"
2070 END PROC
```

****
PROCS MAX AND MIN

These two procedures were sent in by Lars Hult (Goteborg, Sweden). PROC MAX find out which of two numeric variables is the greater, and puts that value into the first variable. So in my example, "8" is printed. Try it with different values. PROC MIN is just the opposite.

```
 10 LET fi=5,se=8
 20 max fi,se
 30 PRINT fi

100 DEF PROC max REF a,b
      LET a=(a AND a>b)+(b AND b>a)
    END PROC

110 DEF PROC min REF a,b
      LET a=(a AND a<b)+(b AND b<a)
    END PROC
```

****
AN ODD FUNCTION

Lars also enclosed a PROC ODD to detect odd numbers, but I have converted it to a function. It is a pity we cannot use a function name like FN odd, as allowed on the SAM Coupe.

```
200 DO
      INPUT x
      PRINT x,FN q(x)
    LOOP

210 DEF FN q(a)=INT (a/2)<>a/2
```

The function gives 1, which is equivalent to "true", when the number in the brackets is odd. It would allow simplification of some programs, such as the file eraser in this issue, which checks for whether a file number is odd or even, so that it knows which of two columns to print the file name in.

****

MONSTERS AND MAZES

I wrote a maze-generation program similar to the procedure at line 1000 quite a few years ago, using a lot of trial and error to make the maze negotiable but not too trivial. Of course, with an overhead view most mazes are pretty easy. More recently I added a "monster" in the shape of a "*" to chase the player, in the shape of a "O". The player uses the Q, S, L and P keys to move - but this can be easily altered by changing line 410.

The "maze number" asked for is the start number for RND, and a given number always gives the same maze. To simply use the previous maze again, press ENTER, and the maze attributes, stored in m$, will be instantly replaced.

The most interesting part of the program to me was controlling the monster - it knows where the player is, but it cannot simply move towards him or it will be trapped in blind alleys. So a list of previous locations of the monster is kept in coded form in a string. PROC movem calculates a "desirability" for a move in each of 4 possible directions, before making the best move. "Desirability" is weighted very heavily towards not walking through the maze walls (!) using ATTR to check this, and is also weighted to moving towards the player, and avoiding previous monster locations. INSTRING searches the string of previous locations and the most recent "previous locations", judged by their position in the string, are avoided most strongly. (If I do not explain this very well it is because it is some time since I wrote the routine, and I've forgotten!) A$ can hold data for 100 previous locations. As each location is visited, its data is JOINed to the end of the string, and the first data in the string (the oldest) is DELETEd. If a location is visited twice, the older entry in the string is set to an odd value so that it is not significant anymore (line 720).

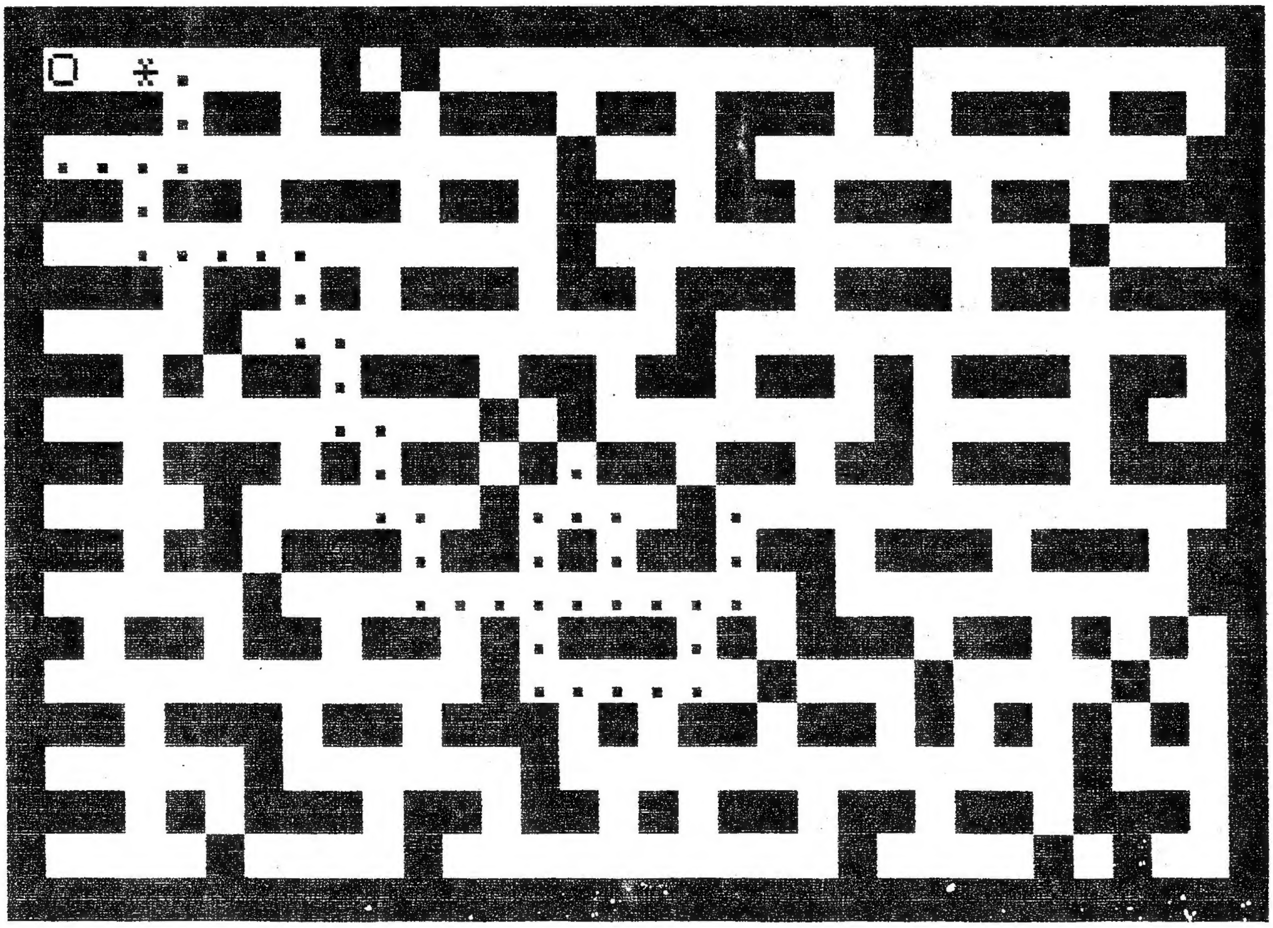
The monster chases the player until he is "got", and the player's score is his survival time. In the first version, line 710 did not use BRIGHT 1, and a space was printed instead of a full stop. The monster did not leave a trail, and sometimes the player could survive indefinitely. The version listed means that this is unlikely to happen, since the monster can cross its own trail but the player cannot.

```
 10 DO
 20   INPUT "Maze number:"; LINE i$
 25   IF i$="" THEN LET m=0
      ELSE LET m=VAL i$
 30   drawmaze m
 40   playgame
 50   kill
 60   DO
      LOOP UNTIL INKEY$=""
 70   PAUSE 0
 80   CLS
      PRINT CSIZE 16;AT 5,3;"SCORE:";sc
 90   PAUSE 0
100 LOOP
```

```
200 DEF PROC playgame
210   POKE 23658,0
      REM lower case
220   DIM p(4)
230   DIM a$(200)
240   LET c=8,g=5,h=5,a=11,b=18,y=a,z=b,k=0,sc=0
250   PRINT AT g,h;"O"
260   PRINT AT a,b;"*"
270   DO
280     movem
290     movep
300     LET sc=sc+1
310   LOOP UNTIL k
320 END PROC

400 DEF PROC movep
410   ON INSTRING(1,"lpsq",INKEY$)+1
      END PROC
      LET q=g+1,r=h
      LET q=g-1,r=h
      LET r=h+1,q=g
      LET r=h-1,q=g
420 IF ATTR (q,r)=56 THEN
      PRINT AT g,h;" ";AT q,r;"O"
      LET g=q,h=r
430 END PROC

600 DEF PROC movem
610   LET e=h-b,f=g-a,d=SGN (ABS e-ABS f),s=SGN f,t=SGN e
620   LET p(1)=(ATTR (a,b+1)<>c)*1000+(t=1)*5-2*INSTRING(1,
      a$,CHR$ (a+128)+CHR$ (b+1))+d*t
630   LET p(2)=(ATTR (a+1,b)<>c)*1000+(s=1)*5-2*INSTRING(1,
      a$,CHR$ (a+129)+CHR$ b)-d*s
640   LET p(3)=(ATTR (a,b-1)<>c)*1000+(t=-1)*5-2*INSTRING(1,
      a$,CHR$ (a+128)+CHR$ (b-1))-d*t
650   LET p(4)=(ATTR (a-1,b)<>c)*1000+(s=-1)*5-2*INSTRING(1,
      a$,CHR$ (a+127)+CHR$ b)+d*s
660   LET mp=p(1),d=1
670   FOR t=2 TO 4
        IF p(t)>mp THEN LET mp=p(t),d=t
680   NEXT t
690   LET y=a,z=b
      ON d
        LET b=b+1
        LET a=a+1
        LET b=b-1
        LET a=a-1
700   IF SCREEN$ (a,b)="O" THEN LET k=1
710   PRINT BRIGHT 1;AT y,z;".";AT a,b;"*"
720   LET t$=CHR$ (y+128)+CHR$ z,d=INSTRING(1,a$,t$)
      IF d THEN LET a$(d)=" "
730   DELETE a$(1 TO 2)
      JOIN t$ TO a$
740 END PROC

900 DEF PROC kill
      PRINT AT y,z;" "
      FOR n=1 TO 12
        PRINT OVER 1;AT a,b;"*"
        BEEP .02,0-n
      NEXT n
    END PROC
```

```
1000 DEF PROC drawmaze m
1010   CLS
1030   IF m=0 THEN POKE 22528,m$
         GO TO 1220
1040   LOCAL a,b,n,p,r,t,z
1050   RANDOMIZE m
1060   LET p=1,a=7,b=p,m=2,n=12
1070   PRINT PAPER p;STRING$(32," ")
1080   FOR r=1 TO 19
1090     PRINT AT r,0; PAPER p;" ";
1100     DO
1110       LET z=RNDM(n)+1,t=0
1120       DO
1130         PRINT PAPER a;" ";
           EXIT IF PEEK 23688<=2
1140         LET t=t+1
           LOOP UNTIL t=z
1150       PRINT PAPER b;" ";
1160     LOOP UNTIL PEEK 23688<=2
1170     PRINT AT r,31; PAPER p;" ";
1180     LET t=a,a=b,b=t,t=m,m=n,n=t
1190   NEXT r
1200   PRINT PAPER p;STRING$(32," ")
1210   LET m$=MEMORY$()(22528 TO 23232)
1220 END PROC
```

Below: One of 65000-odd possible mazes, with occupants. (Some mazes are unplayable due to the player starting off in an isolated pocket of the maze.)



****

PROPORTIONAL-SPACED TEXT JUSTIFICATION

Several readers have expressed an interest in the program I used to print some earlier newsletters using variable-width spaces to give more elegant right-justification. I have delayed doing this, since I wanted to polish the routine a bit (it is called TRJ for Temporary Right-Justify!) but this is my last chance, so here it is. The first thing to say is that it does not deal with proportional-spaced text - it just uses variable-width spaces generated by sending bit-image data, as in a screen dump, but all blank. Unfortunately, the frequent changes to bit-image mode make the program quite slow, which is why I have not used it recently, but at least it works on printers like my RX80 which have no genuine proportional-printing mode. (I have received two versions of a right-justification procedure which works with true proportional text from John Watkins of 8 Hammond House, Tiller Road, London, E14 8PW. His procedures "know" how wide each letter is in proportional mode and can either space the words further apart to fill each line, or space every character more widely. The output looks very good. Unfortunately I cannot test the procedures on my non-proportional printer and this discouraged me from typing them in! Perhaps John could supply a copy to interested readers?) My program is designed to read in Tasword III files and print all or part of them, but it could be modified to deal with Tasword II files instead. Line 70 sets "emphasised" and "left margin 8". Line 80 sets the line length to 64.

```
 10 REM TRJ
 20 CSIZE 4,8
 30 INPUT "file name?";n$
 40 INPUT "first line?";first
 50 INPUT "last line?";last
 60 CLOSE #3
    OPEN #3;"b"
 70 LPRINT CHR$ 27;"E";CHR$ 27;"l";CHR$ 8;
 80 LET LL=64,a$=STRING$(30,CHR$ 0)
 90 LET e$=CHR$ 139+CHR$ 132+CHR$ 140+CHR$ 131+CHR$ 138
100 CLOSE #5
    OPEN #5;"m";1;n$
110 FOR n=1 TO first-1
      INPUT #5; LINE t$
      LET j$=INKEY$#5
      PRINT t$
    NEXT n
120 FOR n=first TO last
130   IF EOF(5) THEN CLOSE #5
        STOP
140   INPUT #5; LINE t$
150   LET j$=INKEY$#5
160   PRINT t$
170   DO
180     IF LEN t$<>LL THEN
           send t$+CHR$ 13+CHR$ 10
190   EXIT IF LEN t$<>LL
200     LET p=1
210     DO UNTIL t$(p)<>" "
           send " "
          LET p=p+1
        LOOP
220     LET b=p,s=0,e=0
```

```
230     DO
          LET p=INSTRING(p,t$," ")
240     EXIT IF p=0
          LET s=s+1,p=p+1
260       DO UNTIL t$(p)<>" "
            LET p=p+1,e=e+1
          LOOP
270     LOOP
280     FOR t=1 TO LEN t$
          LET e=e+(t$(t)>=CHR$ 128)
        NEXT t
290     LET pads=(s+e)*6
300     LET p=b
310     DO
320       IF t$(p)=" " THEN
            LET w=INT (pads/s)
            LET s=s-1,pads=pads-w
            send CHR$ 27+"K"+CHR$ w+CHR$ 0+a$( TO w)
            DO
              LET p=p+1
            LOOP UNTIL t$(p)<>" "
330       send t$(p)
340       LET p=p+1
        LOOP UNTIL p>LEN t$
350     send CHR$ 13+CHR$ 10
360   LOOP UNTIL 1
370 NEXT n
380 CLOSE #5

390 DEF PROC send c$
      FOR t=1 TO LEN c$
400     ON INSTRING(1,e$,c$(t))+1
          LPRINT c$(t);
          LPRINT CHR$ 27;"4";
          LPRINT CHR$ 27;"5";
          LPRINT CHR$ 27;"-";CHR$ 1;
          LPRINT CHR$ 27;"-";CHR$ 0;
          LPRINT CHR$ 27;"E";
410   NEXT t
    END PROC
```

PROC SEND checks characters to be sent to the printer; if they are found in e$, which contains a list of some of the graphics characters used by Tasword to control italics and such, special control sequences are sent instead. Normally, this is done by Tasword as it prints a document, but here we must do it ourselves. If a character is not a control character, it is just LPRINTed. Stream 3 should be OPEN to a "B" type channel.

With my printer, it takes about 17 seconds to print a line of emphasised text with this procedure, or about 15 minutes per page! However, if there are some lines which are not supposed to be right justified, like program lines, these are detected as lines which are shorter than the line length of 64 and are printed in a straight-forward way, which speeds things up.

****

MEMOIRS OF A ROM AUTHOR

Many of you will know that I wrote the Basic for MGT's SAM Coupe computer, and I thought perhaps there might be some interest in a behind-the-scenes look at how this was done. Well, fairly soon after Bruce Gordon described the screen memory layout of the Coupe to me, I got excited by how fast writing data to the screen could be, and wrote subroutines for PLOT, DRAW and CIRCLE which were quite satisfactorily fast. Since I did not have a Coupe, I pretended that 24K of my Spectrum's memory was the Coupe's screen memory and wrote the data to that. An interrupt-driven routine then copied a given quarter of the Coupe data to the Spectrum's screen where it was visible, in a distorted form, as 4-by-1 pixel patterns corresponding to each theoretical colour. In November 1988 the first of the current generation of Coupe prototypes was fired up. This was built on several boards and consisted of 100s of separate chips. Fitted with a Spectrum ROM, it ran my machine-code graphics routines as I hoped it would. Thus encouraged, I wrote more graphics routines and print routines. RECORD and BLITZ came as modifications of routines I had published in the Newsletter to allow graphics commands to be recorded and played back, although of course they are much faster and more convenient on the Coupe.

Every few months, I would take the train to Swansea and test what I had produced on the prototype, although I could not actually modify my programs if they didn't work, since I couldn't bring my roomfull of equipment with me. I had hoped that by early 1989 I might have a machine of my own, but unfortunately the highly skilled labour required to duplicate the single prototype just wasn't available. MGT was and is rather a small company, and unfortunately too many jobs devolved to a very small number of people with the required abilities.

Fairly soon I had problems with memory, because I had used up almost all my Spectrum's RAM with a mini-interpreter (extracted from BB), 24K of Coupe screen, and the keywords I had implemented. Fortunately I remembered a Multiface that I had modified some years before to act as a 16K switched RAM. This could replace the Spectrum ROM and give me 64K of RAM to play with. The transition was a bit tricky, because at this stage the infant Coupe interpreter was very dependent on Spectrum subroutines, and I was also partly using Basic as a debugging tool. From time to time nothing would work and I would have to work out why by sheer head-scratching! Another problem was lack of a disc-drive that would work with the system - I had to resort to tape, although assembled code could be sent from my CPC via RS232 link.

A big milestone was passed when my floating point calculator was able to replace the Spectrum equivalent; another occurred when the new expression evaluator, able to deal with a completely new variable format, finally worked. Gradually I added new commands and functions, taking on more off the job than originally intended, because delays with the hardware meant more time was available. Finally, on 14th. September 1989, I had a working Coupe - the first one built on a real printed circuit board. (Bo Jangeborg was writing a graphics program on the original prototype.) Now I could actually see a display and get things like palette switching and sound and tape output working. However, one thing I couldn't get working on either an unadorned

Coupe or a Spectrum was memory paging; the screen and the Basic program and variables were supposed to be paged in and out as needed, in the same area as the upper half of the ROM. Unfortunately it is not feasible to make a new 32K ROM every 10 minutes, so my ROM code would have to be tested in RAM.

I had written code to control paging which looked as though it should work, but since virtually every function of the ROM depended on untestable bits of program, I was rather concerned! It had been evident from the start that what was needed was an add-on that contained 32K of RAM which could be made to act like the Coupe ROM in terms of paging. There were problems building this device, but it was finally working on the 13th. of October. By then I was under a lot of pressure, because Christmas was approaching, the ROM was nearly full, and the paging required a lot of modifications and shuffling of code from one half of the ROM to the other. Besides, the DOS bootstrap and Network operating code still had to be supplied by MGT and incorporated. In late November and early December I worked at MGT in Swansea, 7 days a week and about 15 hours a day. I lost half a stone and developed a nasty rash and a bad temper. Finally the ROM was pronounced finished, although there really hadn't been as much time for checking as anyone would have liked. Subsequently a number of bugs appeared, mostly to do with incorrect handling of the paging system. These have all been fixed in later ROMs, and the plan is to send free replacements to early purchasers of the machine. As it happened, less than a thousand machines were sent out before Christmas, due to manufacturing problems. This was very unfortunate for MGT, who had tried very hard to satisfy their customers, and had virtually the entire workforce toiling to build power supplies! Besides, Christmas to the computer trade is rather like harvest time for a farmer, and needs to be exploited hard because it is a long lean time till the next one!

Currently I am working on an improved Coupe DOS, which would include some Beta Basic features I had to leave out of the ROM because of lack of space, like ALTER/REF, SORT, INARRAY, USING, etc., as well as some totally new things. With 256K of RAM in the entry-level machine and lots of entry points in the ROM to allow commands to be added, I could keep producing new versions for a long time, provided the financial aspects were OK. I am excited by the possibilities.

****

PART-SCREEN COPY ON THE ZX PRINTER

This little procedure grew out of some correspondence with a Portugese correspondent. To COPY, say, the top 100 scan lines of the screen, use COPI 100. The procedure illustrates a general method of "editing" an unchangeable ROM routine - assign it to a string using MEMORY$, alter it by assigning new character values to some parts of it, then call it using LENGTH to find the start address. Here most of the ROM COPY routine is used, and a RET (CHR$ 201) is needed at the end. The third position in the string normally contains CHR$ 176, for a 176-line COPY; this is fixed in the ROM, but alterable here.

```
10 DEF PROC copi lines
20   DEFAULT lines=176
30   LET a$=MEMORY$()(3756 TO 3806)+CHR$ 201
40   LET a$(3)=CHR$ lines
50   RANDOMIZE USR LENGTH(0,"a$")
60 END PROC
```

****
MULTIPLE SORTS

Beta Basic's SORT command is fairly flexible and fast, but sometimes an array has to be SORTed in complicated ways that make it hard to apply SORT. For example, suppose an array contains strings relating to books in a library. One area of each string (a "field") would probably contain the author's name, making it easy to SORT the books according to author. After this, though, you might well want to SORT the books by each author according to some other field, such as year of publication, or title, while still keeping each author's work together. Jacob Baars of Morenhoven, West Germany, sent a procedure to do this. I have modified it a bit, and added some lines to demonstrate it in action on an array of 50 10-character strings. Positions 1-8 in each string are reserved for names, and positions 9-10 store numbers - pretend they are dates!

The program sets up the array randomly to start with, and then asks for the field boundaries for the main SORT. Here you enter 1 and 8 to sort according to name. Then you have the option to sort according to another field - enter 9 and 10. Then you can just press ENTER to see the final list.

The second SORT is much slower than the first, because the program has to look through the array for sublists in which the previous field is the same throughout, and then SORT each sublist according to the new field.

```
  10 DIM a$(50,10)
  20 FOR n=1 TO 50
  30   ON RNDM(5)+1
         LET a$(n)="SMITH"
         LET a$(n)="JONES"
         LET a$(n)="BROWN"
         LET a$(n)="GORDON"
         LET a$(n)="WHITE"
         LET a$(n)="GREEN"
  40   LET a$(n,9 TO )=USING$("##",RNDM(99))
  50   PRINT a$(n)
     NEXT n
  60 mul_sort a$
  70 FOR n=1 TO 50
       PRINT a$(n)
     NEXT n

1000 DEF PROC mul_sort REF i$
       LOCAL c,x,y,z
       field sa,so
       SORT i$()(sa TO so)
       LET c=0
       DO
         PRINT TAB 10;"sorted!"'" extend sort to more fields(
         y/*)"
         PAUSE 0
       EXIT IF SHIFT$(1,INKEY$)<>"Y"
         LET c=c+1
         IF c>1 THEN LET sa=bg,so=ed
```

```
1010      field bg,ed
          LET x=1
          DO
            LET y=x
            DO
              LET y=y+1
            EXIT IF y> LENGTH(1,"i$")
            LOOP UNTIL i$(y,sa TO so)<>i$(x,sa TO so)
1020        SORT i$(x TO y-1)(bg TO ed)
            LET x=y
          LOOP UNTIL x> LENGTH(1,"i$")
        LOOP
      END PROC

1030 DEF PROC field REF beg, REF end
        INPUT "input field boundaries for sort:";beg,end
      END PROC
```

****

SPECIAL EFFECTS WITH ALTER

G. Burtenshaw (Shifnal, Shropshire) has drawn my attention to some startling screen effects that can be produced by ALTER. As he says, "this could be used as an 'explosion' accompanied by sound effects, although overexposure could damage your eyes!" Try something like this:

```
10 DO
     PAUSE 1
     ALTER PAPER 1 TO PAPER 7
     ALTER PAPER 7 TO PAPER 2
     ALTER PAPER 2 TO PAPER 1
   LOOP
```

New flickering colours are produced in strange patterns. You can also use FLASH and BRIGHT, or alter the position and number of PAUSEs, to vary the effect. (PAUSE synchronises the program with the screen display and keeps patterns in a fixed place.)

****

TIP - ALTER AND VARIABLE NAMES

I sometimes type in a program and then notice that the case of variable names is semi-random because I have been changing in and out of Caps Lock while typing in strings. Or I decide that the use of a lower-case L is going to be confused with the numeral 1 in printouts. In such cases I make use of ALTER, which does not care what case a searched-for variable name is, but replaces with the case you specify. So:

```
ALTER x TO x: ALTER Y TO y
```

alters all uses of the variables x and y to lower case, and:

```
ALTER L TO L
```

alters all uses of the variable L to upper case.

****
MATTERS ARISING

George Baldwin's letter in the last issue asking for details of Microdrive Doctor utilities prompted a number of letters. One reader recommended M/Drive Doctor 3.0 from PIPEQ Systems, 151 Millbridge, Dollis Valley Way, Barnet, Herts., another had used M-DOC from Seven Stars Publishing, 34 Squirrel Rise, Marlow, Bucks, SL7 3PN, another highly recommended RAMDOS UTILITIES from Roybot, Rayleigh, Essex (sorry that's all the address I have). Obviously some of these programs may no longer be available.

Several readers reported that the DATA for the sound procedures in issue 14 did not add up to 9691 as it should, and that the program crashed. G. Jackson (Creigiau, Cardiff) impressed me by actually working out what caused the "re-sounding crash". Sorry about the problem. One flaw in my "translate programs to Tasword files" routine is that long lines like line 510 on page 6 of issue 14 need to be tidied up a bit to indent correctly; in doing this I introduced an extra comma between 1 and 6 in the third row of numbers. The row should read: 217,65,16,254,217 etc. Some copies of the Newsletter were hand-corrected.

This is the LAST Newsletter - there will be no more. The subscriptions of most readers have expired with this issue, but a few people renewed for 6, 9 or even 12 issues after issue 12. If you are one of those, and you didn't get a refund with this Newsletter, please write and ask for one!

****
ADVERTS

GET A FULL INDEX OF CONTENTS OF BB NEWSLETTER, FORMAT, ZX COMPUTING (1984-1987) ETC., CHOICE OF FILE FORMATS.
S.A.E. TO LOU OLIVER, 11 FAIRHILL CRESCENT, PERTH, PH1 1RR FOR DETAILS.

SHARES AND SAVINGS (from Eric Day)

As so many of us have Shares, Unit Trusts, etc. apart from our Savings not forgetting the value of our house, I would like to draw Reader's attention to the very excellent program compiled by Charles Buszard of "Thirteen" Grove Wood Close, Chorleywood, Herts, WD3 5PU.

The program automatically selects 48K or 128K mode, all items are easily Entered, Deleted &/or Updated. Share Holdings, Savings & the Updated Summary totals can be Printed-Out on a Sinclair ZX, an Alphacom or on a full-sized printer.

Charles is more than willing to share his program, which is only available (at the present time) on Microdrive cartridge. However I would suggest that, out of politeness, a Blank cartridge formatted "MISER" plus an extra one for his trouble and a 7.5"*5" stamped addressed envelope be included.

Also please let him know the Version of your Sinclair Interface One. (PRINT PEEK 23729; if it gives zero it is version 1, otherwise it is version 2.) Further, give a general idea of the type of Hardware used; TV or Monitor, Printer type serial or parallel, Epson compatible etc.

****

READERS' LETTERS

Dear Dr. Wright,

I know you are involved with the SAM micro. Will it be compatible with the Opus Discovery disc-drive? I use the drive all the time and have all my files on Opus discs.

Alan Rutherford, Farnborough, Hants.

I am in the same position. I transferred a lot of Newsletter programs by saving them to tape and then translating them with an extended version of "BTRANS", the program provided by MGT to translate Spectrum Basic programs. I also needed to do a bit of fiddling and not all programs were fully translatable. I have tried copying the Spectrum ROM to the Coupe's RAM, and plugging in a Discovery via a "twister board", which I thought ought to work, because the Discovery uses special memory locations for communicating with the disc drive, rather than ports (which could conflict with the Coupe's ports). Unfortunately, it didn't work, and I hesitated to play too much in case my machine was damaged somehow. Perhaps there was a connection missing in the twister board (it worked with my Multiface, though). I have loaded sectors from Discovery discs using the Coupe's disc drive and READ AT, and I think some utility to read disc files and translate them could be written.

Dear Andy,

I have a problem reading port 254. Usually (on my old Spectrum at least), this port would return 255 for an EAR signal, and 191 for no signal, yet all I seem to be able to get is a constant value of 73... I CAN read the port using a small machine code routine.

Antony Legat, Blakedown, Worcs.

I couldn't duplicate this problem on my machine, but I suggested something that Antony later reported worked. That was to use IN 65534 instead of IN 254. It was just a guess, but I knew that the upper byte of a port address goes onto the address lines during IN, and I knew that some resistors that reduce spurious low-voltage signals during IN were omitted in some Spectrums. So I thought maybe doing IN with the upper byte all high might be more reliable (65534 is FFFEH and 254 is 00FEH), and it was. By the way, most people think that the Z80 has just 256 ports, and it is true that usually even less than 256 port addresses are recognised by the hardware design of most Z80 computers. But if the other chippery allows it, the Z80 can deal with 65536 input and output ports; some designs even used this to access each byte in the screen via a port address!

Dear Dr. Wright,

I find long program lines do not indent correctly on my dot-matrix printer when they wrap onto the next line.

P. Harrison, Bedford

Try POKEing 57500 (line length for LLISTs) with, say, 64, so that BB goes to a new line before the printer has to. (The initial value at this address is 80.)

Dear Andy,

Can Beta Basic 3.0 be run on the SAM?

(Several readers)

It can if you load a complete copy of the Spectrum ROM from tape into the Coupe's RAM, which is quite easy. (FORMAT magazine has published details.) Unfortunately you cannot then easily use the disc drive, printer, function keys or superior screen modes. LERM Software, 11 Beaconsfield Close, Whitley Bay, Tyne & Wear, NE25 9UW (091-2533615) publish a utility tape to get round such problems for many Spectrum programs, as well as Beta Basic. However, most BB 3.0 programs translate into SAM Basic quite easily (send me a disc and an S.A.E. if you want a modified copy of BTRANS to help with this) and they will then often run 3 or 4 times faster, and can look much prettier in the new graphics modes. (Pixel-resolution colour is very addictive!)

Dear Andy,

Devastated to hear of imminent demise of BB Newsletter!... Have you considered the effects on the world-wide suicide rate if we subscribers are denied our regular fix of BB news!!?? Are you thinking of starting a similar newsletter for SAM-Basic or leaving that to B. Brenchley (editor of FORMAT) perhaps. Whatever you decide, MANY thanks for Beta Basic and for the hours of please given to me and I'm sure many others. Best of luck in all future enterprises. Regards,

Michael Williams, London

It was probably pretty self-indulgent of me to publish the above, but it was representative of quite a few letters, and brought more tears to my eyes than most! Over the years of publication I have received a small mountain of encouraging, interesting, friendly, helpful etc. letters from a great bunch of readers. This was really the motivation for producing a Newsletter in the first place. I am very sorry that this will be the last Newsletter. And I am sorry it has taken such a long time to produce!! Unfortunately, for some time now I have been employed on projects for other people where deadlines really matter and are hard to meet! The SAM Coupe ROM was rather exceptional, but I am sure the problem will remain. Not that I don't usually love my work (which makes me unusually fortunate); it is just that it is hard to find any time, even for holidays. (My wife wasn't too pleased to have to go on her own last year!) I probably should have given up after issue 12, but I was persuaded otherwise. I will still be here if you are really stuck with some programming problem or want Newsletter back-issues, or have corrupted your only copy of Beta Basic. I wish you all the very best!